(JP-A-5-88887)

[Scope of Claim for a Patent]

[Claim 1]

A data processor comprising first data storing means having a width of n bits (n is an arbitrary integer), second data storing means having a width of m bits (m is an arbitrary integer), an instruction register for holding an instruction word, control means for decoding an instruction in the instruction register to control execution and arithmetic means for executing the instruction, wherein

the control means, when decoding a predetermined instruction, defines, in accordance with a bit number k (k is an arbitrary integer) and a direction designated by instruction word information, low-order k bits of the first data storing means as a high-order group and all bits of the second data storing means as a low-order group and shifts linked data of the high-order group and the low-order group by k bits in the direction, so that high-order k bits as the result are stored in low-order k bits of the first data storing means and remaining low-order bits as the result are stored in the second data storing means.

[Claim 2]

A data processor according to Claim 1, being realized in one chip.

[Detailed Description of the Invention]

[0001]

[Industrial Field of Utilization]

The present invention relates to a data processor and more particularly to a data processor having instructions for shifting data in accordance with predetermined data size, direction and bit number designated within the data size in a microprocessor or the like.

[0002]

[Prior Art]

Recently, even a small instruction processor such as a microprocessor has abundant functions with the progress in the advanced integration technique of semiconductor integrated circuits. For example, Z80 of Zilog company is a 8-bit microprocessor but has abundant instruction sets. Z80 is now taken as an example and RLD and RRD instructions of BCD type shift instructions for the unit of 4 bits belonging to one instruction set of Z80 are described to thereby point out problems.

[0003]

The RLD and RRD instructions are instructions for shifting data 4 bits right and left at a time and act on contents of an address in a memory indicated by an A register and an HL pair register. These instructions are effective in operation of BCD type data. Operation of these instructions is illustrated in Figs. 8(1) and (2) in brief. That is, low-order 4 bits of the A register of 8 bits are defined as a high-order group and high-order 4 bits and low-order 4 bits of 8-bit data of an address indicated by the HL pair register are defined as a low-order group. Linked data of the high-order group and the low-order group are rotated right and left in 4-bit unit. At this time, the data size of data of the A register and data of the address indicated by the HL pair register is fixed to be 8 bits and the number of bits to be rotated is also fixed in the 4-bit unit.

[0004]

It is supposed that, for example, data of the octal number are stored in n-th to (n+m)-th addresses of a memory and the operation that the data stored in the n-th to (n+m)-th addresses are shifted 3 bits wholly is performed by the instruction of Z80.

[0005]

As described above, since the rotation instruction makes rotation in the 4-bit unit, an SLA (arithmetic shift) instruction is first executed to shift "0" from the least significant bit of 8-bit data in the n-th address, so that data shifted out from 7-th bit of the A register is stored in a carry flag. An RL (rotate left) instruction is then executed to rotate linked data of the carry flag to the most significant bit of 8-bit data in the n+1st address left. Consequently, the most significant bit of 8-bit data in the n+1st address is shifted to be stored in the carry flag. Next, the RL instruction is executed once more to rotate linked data of the carry flag to the most significant bit of the n+2nd address left. When the above operation is repeated until (n+m)-th address is reached, data in the n-th to (n+m)-th addresses are shifted one bit wholly. Accordingly, in order to shift data three bits, the above operation must be repeated three times, so that a large number of instructions and cycles are required.

[0006]

[Problem to be solved by the Invention]

As described above, the conventional data processor has a drawback that when continuous data are shifted by arbitrary bits, a large number of instructions and cycles are required and the processing time is increased.

[0007]

The present invention is to solve the above problem and an object of the present invention is to provide a data processor in which an instruction for shifting data in accordance with a predetermined data size, direction and bit number designated within the data size is provided and which processes operation for shifting continuous data by arbitrary bits at high speed.

[0008]

[Means for Solving Problem]

In order to solve the above problem, as shown in Fig. 1, the feature of the present invention resides in that the data processor comprises first data storing means 1 having a width of n bits (n is an arbitrary integer), second data storing means 3 having a width of m bits (m is an arbitrary integer), an instruction register 5 for holding an instruction word, control means 7 for decoding an instruction in the instruction register 5 to control execution and arithmetic means 9 for executing the instruction. The control means 7, when decoding a predetermined instruction, defines, in accordance with a bit number k (k is an arbitrary integer) and a direction designated by instruction word information, low-order k bits of the first data storing means 1 as a high-order group and all bits of the second data storing means 3 as a low-order group and shifts linked data of the high-order group and the low-order group by k bits in the designated direction. High-order k bits as the result are stored in low-order k bits of the first data storing means 1 and remaining low-order bits as the result are stored in the second data storing means 3.

[0009]

The second feature of the present invention resides in that in the data processor as set forth in claim 1, the data processor is realized in one chip.

[0010]

[Operation]

In the data processor of the present invention, when a predetermined instruction is set in the instruction register 5, the control means 7 defines, in accordance with a bit number k (k is an arbitrary number) and a direction designated by instruction word information, low-order k bits of the first data storing means 1 as a high-order group and all bits of the second data storing means 3 as a low-order group and shifts linked data of the high-order group and the low-order group by k bits in the direction. High-order k bits as the result are stored in low-order k bits of the first data storing means 1 and remaining low-order bits as the result are stored in the second data storing means 3.

[0011]

Thus, the instruction for shifting data in accordance with the bit number k and the direction designated by the instruction word information is provided and the operation for shifting continuous data by arbitrary bits can be processed at high speed.

[0012]

[Embodiment]

An embodiment according to the present invention is now described with reference to the accompanying drawings.

[0013]

Fig. 2 is a schematic diagram illustrating a data processor according to an embodiment of the present invention.

[0014]

The data processor of the embodiment comprises a microprocessor 10 (hereinafter abbreviated to MPU) of 16 bits and a memory 30 in which each word is constituted by 16 bits, and the microprocessor 10 and the memory 30 are connected to each other through a data bus 27 and an address bus 29.

[0015]

The MPU 10 comprises a bus interface unit 13 (hereinafter abbreviated to BIU) functioning as an interface to the memory 30 and including a data latch DR, an address latch AR and temporary registers TR0

· and TR1, an instruction register group 5 including four instruction registers IR1 to IR4, an instruction decoder 6 for decoding an instruction supplied from the instruction register group 5, a general register group 11 including a bank pointer BP, a various-stack pointer SP and register groups R0 to R15, a constant ROM 15 for storing constants used in arithmetic operations, an arithmetic unit 9 for executing various arithmetic operations and shift registers SR1 and SR2 connected to an input side of the arithmetic unit 9 and used in shift instructions and the like. The above constituent elements make transmission and reception of data through an X bus 21, a Y bus 23 and a Z bus 25. Further, in Fig. 2, thick lines represent buses or signal lines for making transmission and reception of data and thin lines represent signal lines for control signal and the like.

[0016]

The data processor of the embodiment executes each instruction as follows:

[0017]

First of all, an instruction is fetched from the memory 30 through the data bus 27 to be inputted into the data latch DR in the BIU 13. The fetched instruction is then taken in a queue of the instruction register 5. In this connection, the instruction register group 5 constitutes the queue by the four instruction registers IR1 to IR4 and this means that when there is no bus access an instruction is fetched ahead to be taken in the queue, so that data processing is made at high speed. Next, the instruction decoder 6 takes in an instruction from the queue of the instruction register group 5 in sequence to decode the instruction and sends its information to controller 7. The controller 7 sends a control signal to each constituent element on the basis of its information and executes the instruction. The controller 7 receives signals from the arithmetic unit 9, the BIU 13 and other constituent elements and further returns a control signal to each constituent element. In addition, the controller 7 controls a program status word (PSW) containing a condition code (CC) through X bus 21, Y bus 23 and Z bus 25.

[0018]

Referring now to Fig. 3, specification of shift instructions executed in the embodiment are described.

[0019]

An RML (Rotate Multibit Left) instruction is first described. It is supposed that the shift bit width designated by the instruction word is k bits. A general register Rn (n=0 to 15) is used as the first data storing means 1 and a general register Rm (m=0 to 15) or a specific address of the memory 30 is used as the second data storing means 3. Low-order k bits of the general register Rn and all bits of the general register Rm or the memory 30 are linked and rotated k bits left. The specific address of the memory 30 is supposed to be designated by a general register Rj (j=0 to 15 and j≠n), for example.

[0020]

Further, an RMR (Rotate Multibit Right) instruction rotates data by designated k bits right similarly to the RML instruction.

[0021]

Operation at the time that the RML or RMR instruction is executed is described with reference to Fig. 4. Fig. 4 illustrates the shift operation in shift registers SR1 and SR2 and processing operation of the arithmetic unit 9 at the time that the RML instruction (shift 10 bits) is executed in the case where a general register R0 is used as the first data storing means 1 and a specific address (address m) of the memory is used as the second data storing means 3.

[0022]

First, data in the general register R0 is transferred to the shift register SR1 through the X bus 21 (refer to (1) of Fig. 4). At the same time, data corresponding to the address m outputted from the address latch AR is transferred to the shift register SR2 through the data bus 27 and the Y bus 23 (refer to (2) of Fig. 4). Next, the data in the shift register SR1 is shifted 10 bits right as shown in (3) of Fig. 4. Further, the shift registers SR1 and SR2 are linked to each other and contents in the linked shift registers SR1 and SR2 are shifted 10 bits left. Consequently, contents in the shift registers SR1 and SR2 are as shown in (4) and (5) of Fig. 4. Data in the shift registers SR1 and SR2 are temporarily stored in the temporary registers TR0 and TR1 in the BIU 13.

[0023]

Next, data of the designated bit width having low-order 10 bits being "1" and remaining high-order bits being "0" is transferred from the constant ROM 15 to the arithmetic unit 9 and data in the general register R0 is transferred to the arithmetic unit 9. The arithmetic unit 9 calculates a logical product (AND) thereof as shown in (6) of Fig. 4. Further, as shown in (7) of Fig. 4, the arithmetic unit 9 calculates a logical sum (OR) of the resultant logical product and the data of the shift register SR2 stored temporarily in the temporarily register TR1.

[0024]

The resultant logical sum is stored in the address m of the memory 30 and the data of the shift register SR1 stored temporarily in the temporary register TR0 is transferred in the general register R0.

[0025]

The RML instruction is executed as described above and the RMR instruction is also executed similarly.

[0026]

Next, an application example in the case where the data processor of the embodiment is used concretely is described.

[0027]

Fig. 5 illustrates a first application example. The general register Rn is used as the first data storing means 1 and the memory 30 is used as the second data storing means 3. Data of low-order 10 bits in general register Rn is inserted into low-order 10 bits of data in address m of the memory 30 and data of high-order 10 bits in address m is shifted to low-order 10 bits in address m+2. Thus, data in addresses of memory are successively shifted to subsequent addresses.

[0028]

First, in (1) of Fig. 5, data in general register Rn (data A is stored in low-order 10 bits) and data in address m are rotated 10 bits left by means of RML instruction. Data B of high-order 10 bits in address m is stored in general register Rn as shown in (2) of Fig. 5. Next, data in general register Rn and address m+2 are rotated 10 bits left by means of RML instruction. At this time, data C of high-order 10 bits in address m+2 is stored in general register Rn as shown in (3) of Fig. 5. Similarly, data in general register Rn and address m+4 are rotated 10 bits left by means of RML instruction, so that data D of high-order 10 bits in address m+4 is stored in general register Rn as shown in (4) of Fig. 5. Finally, data in general register Rn and address m+6 are rotated 10 bits left by means of RML instruction, so that a final state as shown in (5) of Fig. 5 is reached. As described above, the continuous shift operation can be made only by repeating the same instruction.

[0029]

Fig. 6 is a diagram illustrating operation of a second application example. In this example, the general register Rn is used as the first data storing means 1 and the memory 30 is used as the second data storing means 3. Low-order 10 bits of data in address m+6 of memory 30 are inserted into low-order 10 bits of general register Rn and low-order 10 bits of data in address m+6 is shifted to high-order 10 bits of address m+4. Thus, data in addresses of memory are successively shifted to next addresses ahead.

[0030]

The operation of the second application example is the same with the exception that the RMR instruction is used to rotate data 10 bits right. As described above, the RMR instruction is effective in the case where data is shifted from lower addresses to upper addresses.

[0031]

Fig. 7 illustrates operation of a third application example. In this example, the general register Rn is used as the first data storing means 1 and continuous general registers Rm to Rm+3 are used as the second data storing means 3. Low-order 10 bits of data in general register Rn are inserted into low-order 10 bits of data in general register Rm and high-order 10 bits of data in general register Rm are shifted to low-order 10 bits of data in general register Rm+1. Thus, data in general registers are successively shifted to subsequent general registers.

[0032]

In (1) of Fig. 7, when data A of low-order 10 bits in general register Rn and data in general register Rm are rotated 10 bits left by means of RML instruction, data B of high-order 10 bits in general register Rm is stored in general register Rn as shown in (2) of Fig. 7. Next, data B in general register Rn and data in general register Rm+1 are rotated 10 bits left by means of RML instruction. At this time, data C of high-order

10 bits in general register Rm+1 is stored in general register Rn as shown in (3) of Fig. 7. Similarly, data C in general register Rn and data in general register Rm+2 are rotated 10 bits left by means of RML instruction, so that data D of high-order 10 bits in general register Rm+2 is stored in general register Rn as shown in (4) of Fig. 7. Finally, data in general register Rn and data in general register Rm+3 are rotated 10 bits left by means of RML instruction, so that a final state as shown in (5) of Fig. 7 is reached.

[0033]

Further, RMR instruction can be used in the same manner as the second application example to shift data in continuous general registers.

[0034]

As described above, in the shift operation between the first and second data storing means 1 and 3 in the data processor of the embodiment, low-order bits for the designated shift bit number of data in the second data storing means 3 are stored in low-order bits of the first data storing means 1 in case of the right rotation and high-order bits for the designated shift bit number of data in the second data storing means 3 are stored in low-order bits of the first data storing means 1 in case of the left rotation. This stored data can be used as shift data in the next rotation instruction. Accordingly, the shift operation of data in continuous memories or general registers can be made only by repeating the same instruction. For example, when any word or phrase is inserted in the sentence processing of a word processor, the operation as the above-mentioned application example is made in the assembler instruction level and a large number of instructions and cycles are required heretofore, although it can be processed at high speed with the reduced number of instructions.

[0035]

Further, the data size can be set arbitrarily and the shift bit number can be designated arbitrarily as far as it is smaller than or equal to the data size.

[0036]

[Effects of the Invention]

As described above, according to the present invention, since the instruction set is equipped with the instruction for defining low-order k bits in first data storing means as a high-order group and data in second data storing means as a low-order group in accordance with a bit number k and a direction designated by instruction word information and shifting linked data thereof k bits in the designated direction to store high-order k bits as the result into low-order k bits of the first data storing means and remaining low-order bits into the second data storing means, there can be provided the data processor which can process the operation for shifting data in any bit unit in any direction and shifting continuous data by any bit number at high speed.

[Brief Description of the Drawings]

[Fig. 1]

Diagram illustrating the principle of the present invention.

[Fig. 2]

Diagram illustrating a data processor according to an embodiment of the present invention.

[Fig. 3]

Diagram illustrating specification of a shift instruction in the embodiment of the present invention.

[Fig. 4]

Diagram illustrating execution processes of RML and RMR instructions in the embodiment of the present invention.

[Fig. 5]

Diagram illustrating operation explaining a first application example of the embodiment of the present invention.

[Fig. 6]

Diagram illustrating operation explaining a second application example of the embodiment of the present invention.

[Fig. 7]

Diagram illustrating operation explaining a third application example of the embodiment of the present invention.

[Fig. 8]

Diagram illustrating specification of a shift instruction in a conventional microprocessor (Z80).

[Description of Reference Numerals]

1	first data storing means
3	second data storing means
5	instruction register (instruction register group)
6	instruction decoder
7	control means (controller)
9	arithmetic means (arithmetic unit)
10	microprocessor (MPU)
11	general register group
13	bus interface unit (BIU)
15	constant ROM
21	X bus
23	Y bus
25	Z bus
27	data bus

29	address bus
30	memory
DR	data latch
AR	address latch
TR0, TR1	temporary register
IR1 to IR4	instruction register
BP	bank pointer
SP	various-stack pointer
R0 to R15	general register
SR1, SR2	shift register

下位ビットを第２のデータ格納手段に格納する命令を命令セットに用意したので、任意のビット単位で任意の方向に桁移動することができ、連続するデータを任意のビット数だけ桁移動する操作等を高速に処理しうるデータ処理装置を提供することができる。

【図面の簡単な説明】

【図１】本発明の発明原理図である。

【図２】本発明の一実施例に係るデータ処理装置の構成図である。

【図３】本発明の実施例における桁移動命令の仕様説明図である。

【図４】本発明の実施例におけるＲＭＬ命令またはＲＭＲ命令の実行過程を説明する図である。

【図５】本発明の実施例の第１の適用例を説明する動作説明図である。

【図６】本発明の実施例の第２の適用例を説明する動作説明図である。

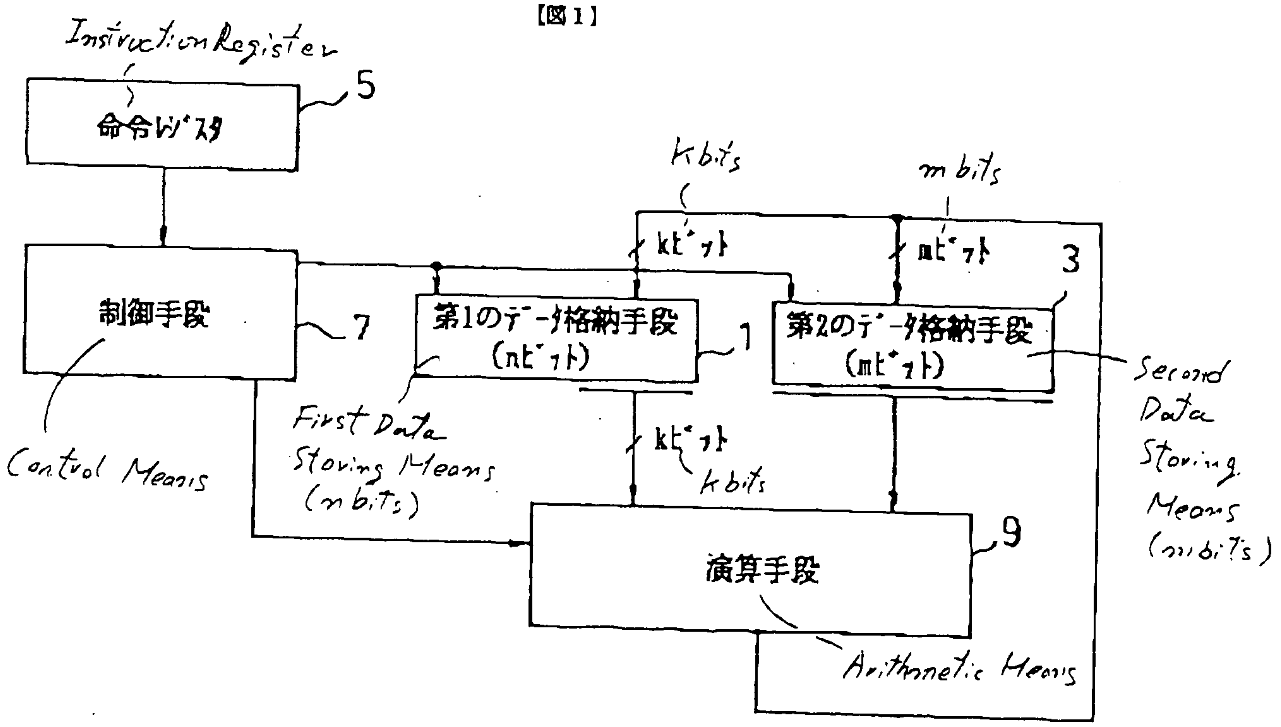
【図７】本発明の実施例の第３の適用例を説明する動作説明図である。

【図８】従来のマイクロプロセッサ（Ｚ８０）における桁移動命令の仕様説明図である。

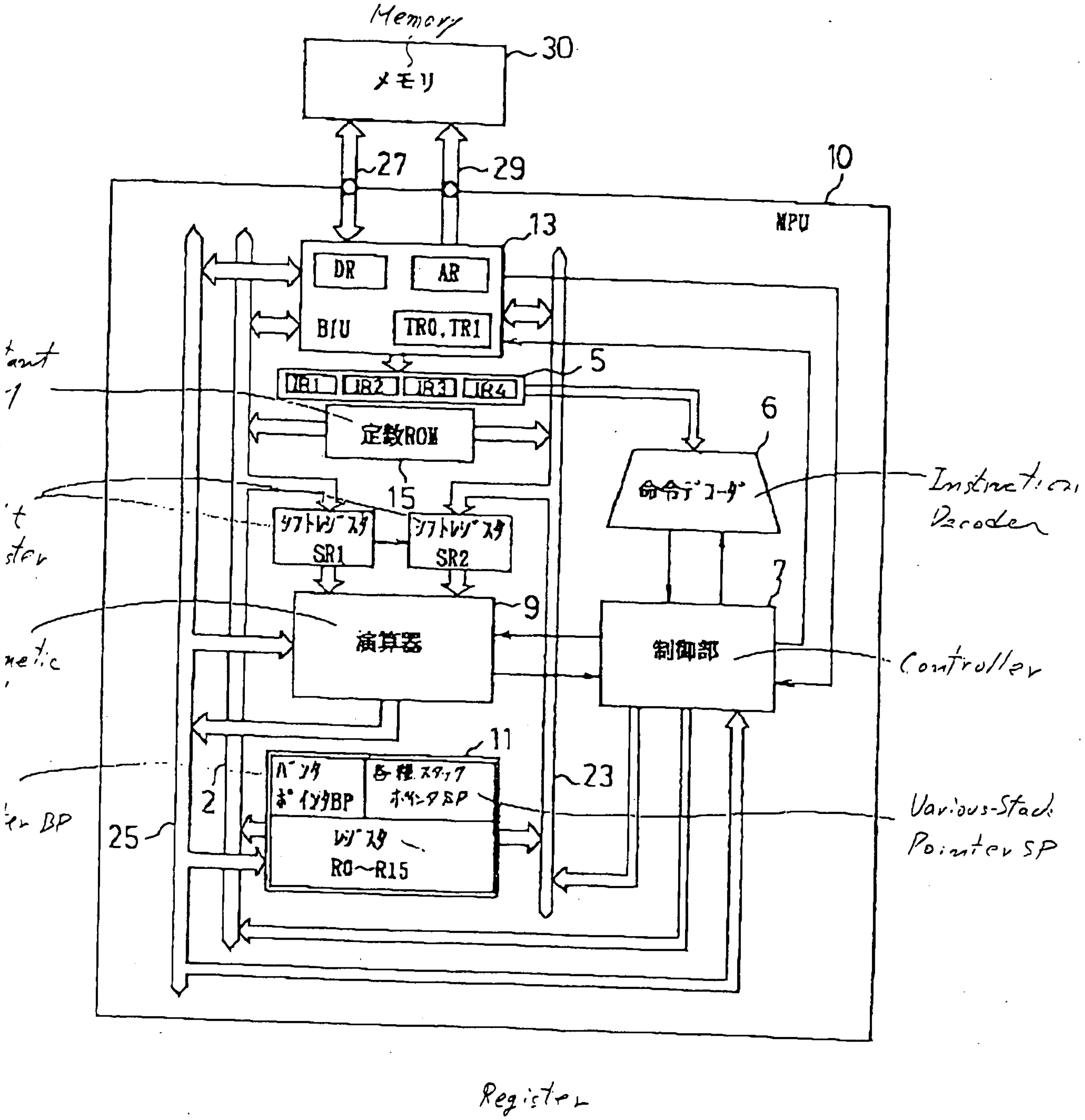
【符号の説明】

１	第１のデータ格納手段
３	第２のデータ格納手段
５	命令レジスタ（命令レジスタ群）
６	命令デコーダ
７	制御手段（制御部）
９	演算手段（演算器）
１０	マイクロプロセッサ（ＭＰＵ）
１１	汎用レジスタ群
１３	バスインタフェースユニット（ＢＩＵ）
１５	定数ＲＯＭ
２１	Ｘバス
２３	Ｙバス
２５	Ｚバス
２７	データバス
２９	アドレスバス
３０	メモリ
ＤＲ	データラッチ
ＡＲ	アドレスラッチ
ＴＲ０，ＴＲ１	テンポラリレジスタ
ＩＲ１～ＩＲ４	命令レジスタ
ＢＰ	バンクポインタ
ＳＰ	各種スタックポインタ
Ｒ０～Ｒ１５	汎用レジスタ
ＳＲ１，ＳＲ２	シフトレジスタ

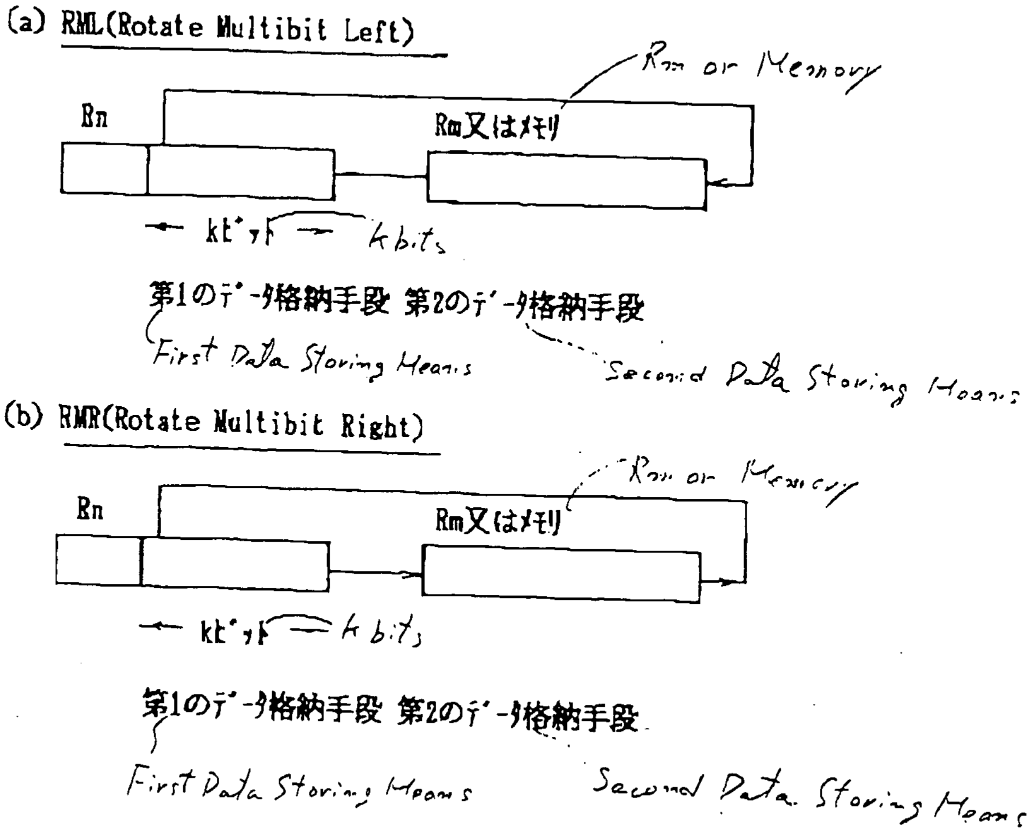
【図１】



【図2】

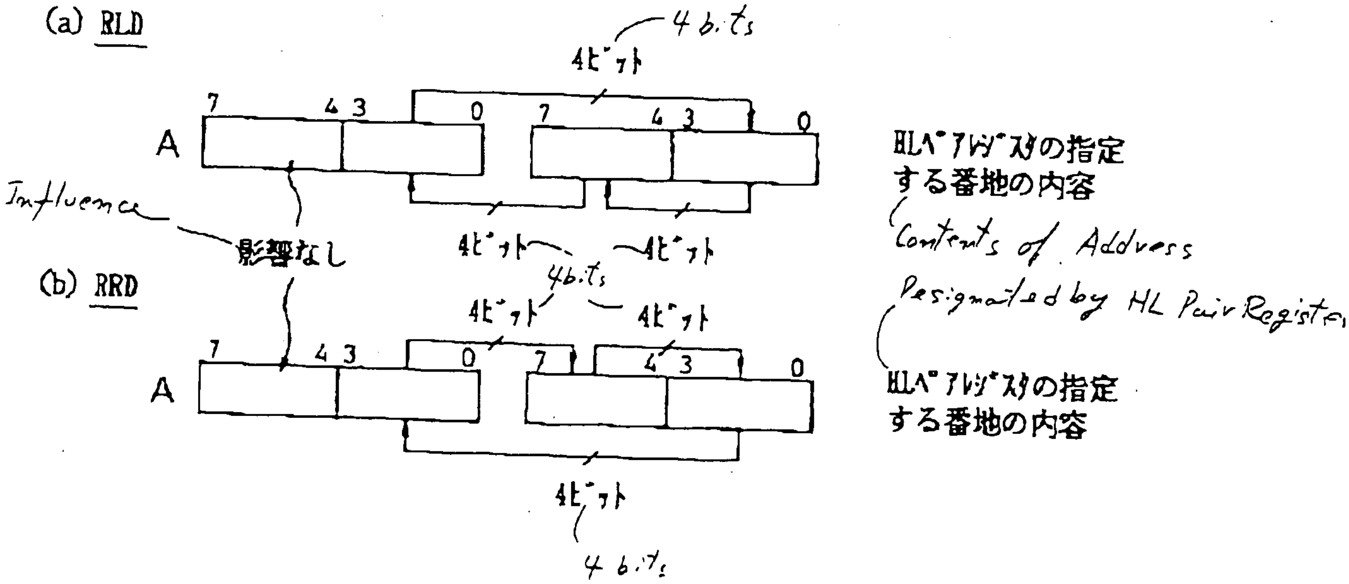

Memory
メモリ
30
27
29
10
MPU
13
DR
AR
BIU
TR0,TR1
IR1
IR2
IR3
IR4
5
tant
1
定数ROM
15
6
命令デコーダ
Instruction
Decoder
t
ster
シフトレジスタ
SR1
シフトレジスタ
SR2
9
演算器
7
制御部
Controller
netic
11
バンク
ポインタBP
各種スタック
ポインタSP
Various-Stack
Pointer SP
ter BP
2
23
25
レジスタ
R0～R15
Register

【図3】


(a) RML(Rotate Multibit Left)

(b) RMR(Rotate Multibit Right)

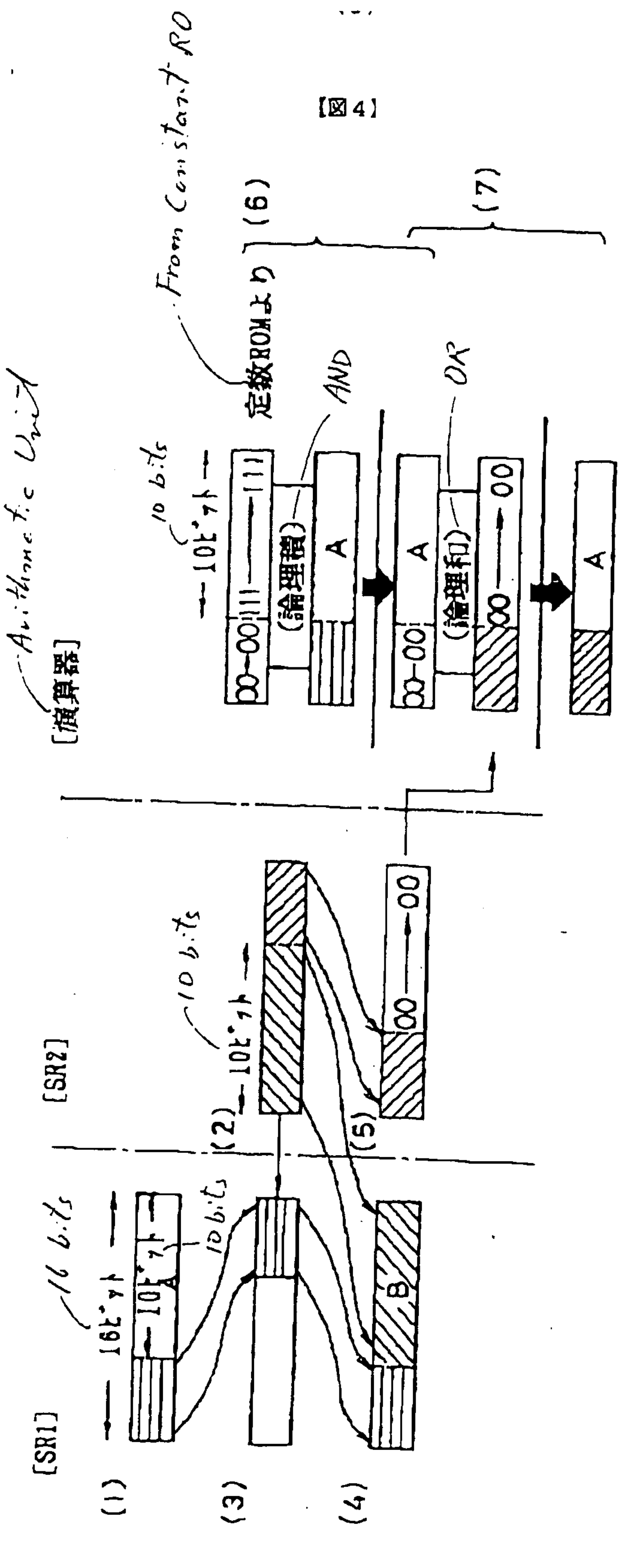
【図8】


(a) RLD

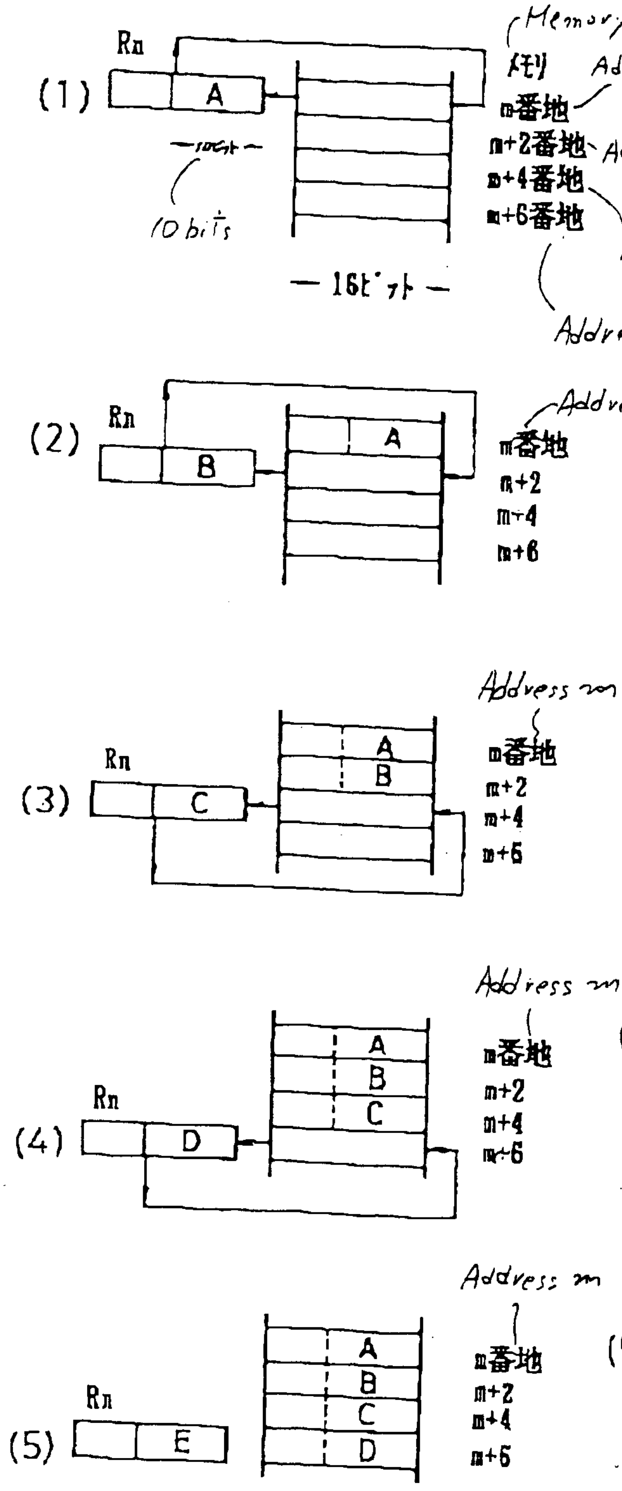
(b) RRD

4ビット
4 bits
7
4 3
0
A
HLペアレジスタの指定する番地の内容
Contents of Address Designated by HL Pair Register
影響なし
Influence

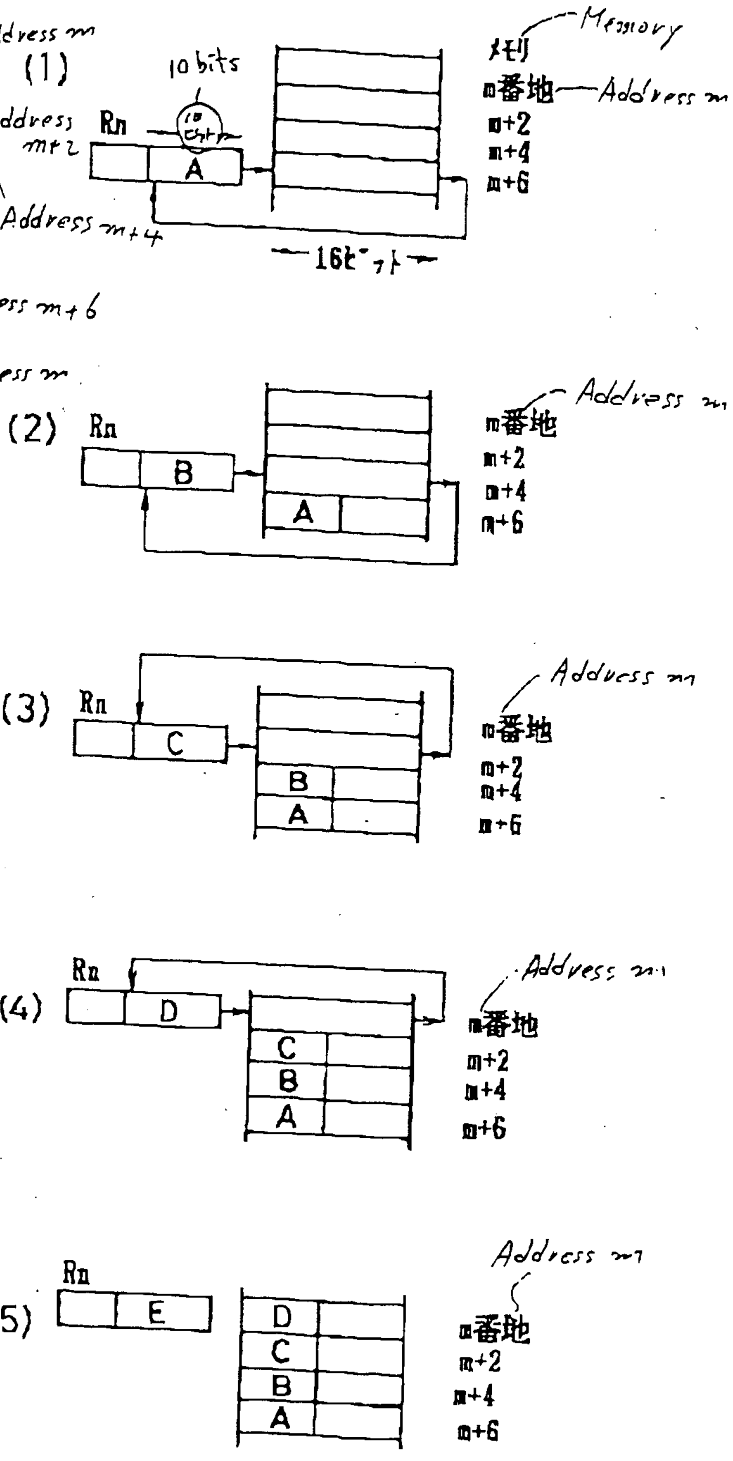
【図4】


[SR1]
[SR2]
[演算器]
Arithmetic Unit
(1)
(2)
(3)
(4)
(5)
(6)
(7)
16ビット
16 bits
10ビット
10 bits
10ビット
10 bits
10ビット
10 bits
A
B
A
A
A
00
00－00
00－00111
111
00－00
00－00
00
00
(論理積)
AND
(論理和)
OR
定数ROMより
From Constant RO

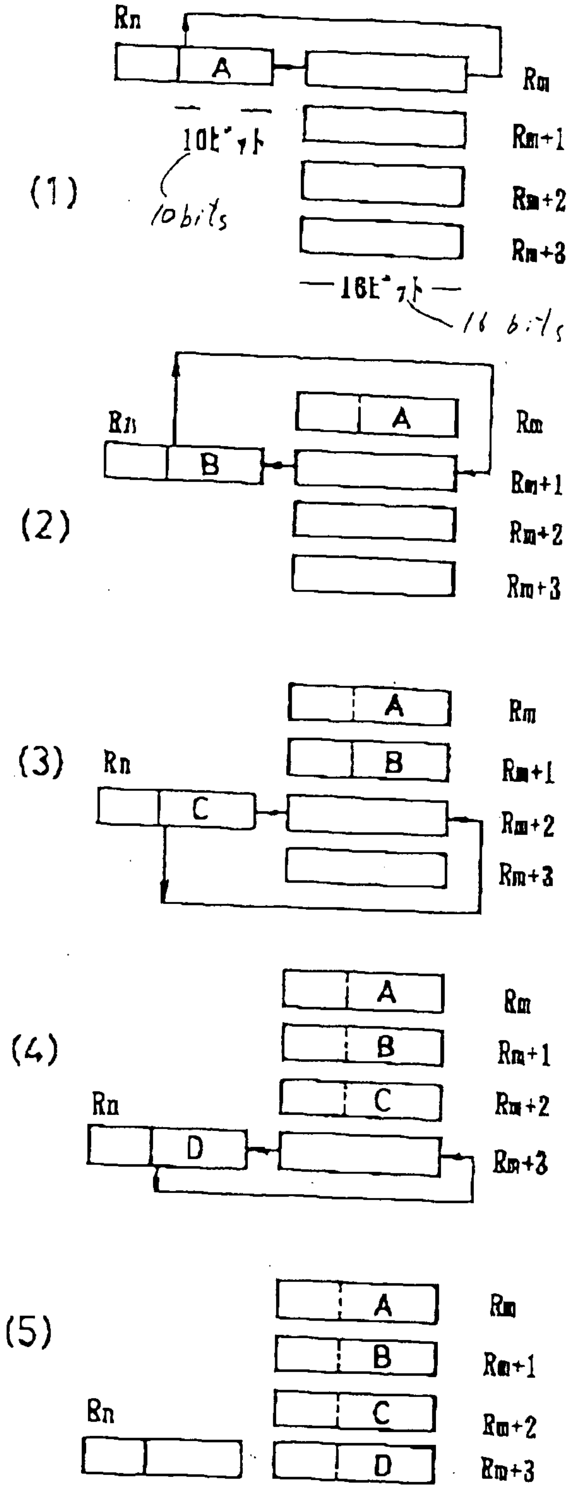
【図５】


(1)
Rn
A
10ビット
10 bits
メモリ
Memory
m番地
Address m
m+2番地
Address m+2
m+4番地
Address m+4
m+6番地
Address m+6
16ビット
(2)
Rn
B
A
m番地
Address m
m+2
m+4
m+6
(3)
Rn
C
A
B
m番地
Address m
m+2
m+4
m+6
(4)
Rn
D
A
B
C
m番地
Address m
m+2
m+4
m+6
(5)
Rn
E
A
B
C
D
m番地
Address m
m+2
m+4
m+6

【図６】


(1)
10 bits
10ビット
Rn
A
メモリ
Memory
m番地
Address m
m+2
m+4
m+6
16ビット
(2)
Rn
B
A
m番地
Address m
m+2
m+4
m+6
(3)
Rn
C
B
A
m番地
Address m
m+2
m+4
m+6
(4)
Rn
D
C
B
A
m番地
Address m
m+2
m+4
m+6
(5)
Rn
E
D
C
B
A
m番地
Address m
m+2
m+4
m+6

【図７】


Rn
A
Rm
Rm+1
Rm+2
Rm+3
10ビット
10bits
16ビット
16 bits
(1)
Rn
B
A
Rm
Rm+1
Rm+2
Rm+3
(2)
(3)
Rn
A
B
C
Rm
Rm+1
Rm+2
Rm+3
(4)
Rn
A
B
C
D
Rm
Rm+1
Rm+2
Rm+3
(5)
Rn
A
B
C
D
Rm
Rm+1
Rm+2
Rm+3